HEWLETT-PACKARD COMPANY
Intellectual Property Administration
P. O. Box 272400
Fort Collins, Colorado 80527-2400

PATENT APPLICATION

ATTORNEY DOCKET NO. 10991880-1

IN THE
UNITED STATES PATENT AND TRADEMARK OFFICE

Inventor(s):	Pratap SUBRAHMANYAM, et al.	**Confirmation No.:**	1056
Application No.:	09/898,351	**Examiner:**	Vu, T. A.
Filing Date:	07/03/01	**Group Art Unit:**	2124
Title:	A SYSTEM AND METHOD TO DECREASE PROGRAM ANALYSIS OVERHEAD		

Mail Stop Appeal Brief-Patents
Commissioner For Patents
PO Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPEAL BRIEF

Sir:

Transmitted herewith is the Appeal Brief in this application with respect to the Notice of Appeal filed on 05/16/05.

The fee for filing this Appeal Brief is (37 CFR 1.17(c)) $500.00.

(complete (a) or (b) as applicable)

The proceedings herein are for a patent application and the provisions of 37 CFR 1.136(a) apply.

( ) (a) Applicant petitions for an extension of time under 37 CFR 1.136 (fees: 37 CFR 1.17(a)-(d) for the total number of months checked below:

( )	one month	$120.00
( )	two months	$450.00
( )	three months	$1020.00
( )	four months	$1590.00

( ) The extension fee has already been filled in this application.

(X) (b) Applicant believes that no extension of time is required. However, this conditional petition is being made to provide for the possibility that applicant has inadvertently overlooked the need for a petition and fee for extension of time.

Please charge to Deposit Account **08-2025** the sum of **$500.00**. At any time during the pendency of this application, please charge any fees required or credit any over payment to Deposit Account 08-2025 pursuant to 37 CFR 1.25. Additionally please charge any fees to Deposit Account 08-2025 under 37 CFR 1.16 through 1.21 inclusive, and any other sections in Title 37 of the Code of Federal Regulations that may regulate fees. A duplicate copy of this sheet is enclosed.

(X) I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450. Date of Deposit: **06/15/05**

OR

( ) I hereby certify that this paper is being transmitted to the Patent and Trademark Office facsimile number _____ on _____

Number of pages:

Typed Name: **Desiree Reardon**

Signature: Desiree Reardon

Respectfully submitted,

Pratap SUBRAHMANYAM, et al.

By _____

John P. Wagner, Jr.
Attorney/Agent for Applicant(s)
Reg. No. **35,398**

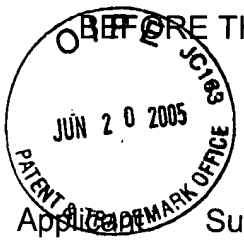
Date: **06/15/05**

Telephone No.: **(408) 938-9060**

AF

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES



Applicant:	Subrahmanyam et al.	Patent Application	
Serial No.:	09/898,351	Group Art Unit:	2193
Filed:	July 03, 2001	Examiner:	Vu, Tuan A.

For: A SYSTEM AND METHOD TO DECREASE PROGRAM ANALYSIS OVERHEAD

Appeal Brief

06/21/2005 TBESHAH1 00000038 082025 09898351
01 FC:1402 500.00 DA

10991880-1

Table of Contents

Real Party in Interest

The assignee of the present invention is Hewlett-Packard Company.

Related Appeals and Interferences

There are no related appeals or interferences known to the Appellant.

Status of Claims

Claims 1-2, 4-12, 14-17 and 19-20 are pending. Claims 7-9 are rejected under 35 U.S.C. § 112. Claims 1-2, 4-12, 14-17 and 19-20 are rejected under 35 U.S.C. § 103(a). Rejections of Claims 1-2, 4-12, 14-17 and 19-20 are herein appealed.

Status of Amendments

All proposed amendments have been entered. An amendment subsequent to the final rejection has been entered. An amendment subsequent to the Advisory Action has not been filed.

Summary of Claimed Subject Matter

In accordance with Independent Claim 1, one embodiment of the present claimed invention pertains to a method to analyze a computer program that includes a plurality of blocks of code. As described in Figure 3 item 27 and beginning on paragraph [0036] of the Specification, a block of code is received to

a code cache. A counter is used for tracking each time the block of code is executed on the code cache, as described in Figure 3, item 24 and described in the Specification beginning at paragraph [0037]. A counter cache is maintained for storing each counter of the block of code while the block of code is stored on the code cache, as shown in Figure 3, item 25 and described in the Specification beginning at paragraph [0037]. A storage area is also maintained for storing each counter of the block of code previously executed on the code cache after the block of code is evicted from the code cache, as shown in Figure 4, item 64 and described in the Specification beginning at paragraph [0044].

The method to analyze a computer program that includes a plurality of blocks of code further comprises identifying when the code cache is full, as shown in Figure 4, item 54 and described in the Specification beginning at paragraph [0040].

The method to analyze a computer program that includes a plurality of blocks of code further comprises determining which the counter of the block of code stored on the counter cache is least recently executed and evicting the least recently executed block of code, related to the counter, from the code cache, as shown in Figure 4, item 55 and described in the Specification beginning at paragraph [0040]. In addition, the counter of the least recently executed block of code is copied from the counter cache to the storage area when the least recently executed block of code related to the

counter is evicted from the code cache, as shown in Figure 8, item 136 and described in the Specification beginning at paragraph [0051].

The method to analyze a computer program that includes a plurality of blocks of code further comprises checking the storage area to determine if the block of code is being executed for other than the first time, as shown in Figure 4, item 51 and described in the Specification beginning at paragraph [0038]. The counter associated with the block of code being executed for other than the first time is then loaded from the storage area into the counter cache, as shown in Figure 4, item 52 and described in the Specification beginning at paragraph [0038]. The counter associated with the block of code being executed for other than the first time is then updated, as shown in Figure 4, item 61 and described in the Specification beginning at paragraph [0043].

In accordance with Independent Claim 6, one embodiment of the present claimed invention pertains to a system for analyzing a computer program that includes a plurality of blocks of code. As described in Figure 3 and beginning on paragraph [0036] of the Specification, a means for executing the computer program is provided. A means for counting counts each time one of the pluralities of blocks of code is executed, as described in Figure 3, item 24 and described in the Specification beginning at paragraph [0037]. A means for maintaining a counter cache is provided for storing the counting means of the plurality of blocks of code that are most recently executed, as shown in Figure 3,

item 25 and described in the Specification beginning at paragraph [0037]. A means for maintaining a storage area is also provided for storing the counting means of the block of code most recently executed, as shown in Figure 4, item 64 and described in the Specification beginning at paragraph [0044].

The system for analyzing a computer program that includes a plurality of blocks of code further comprises a means for identifying when the code cache is full, as shown in Figure 4, item 54 and described in the Specification beginning at paragraph [0040].

The system for analyzing a computer program that includes a plurality of blocks of code further comprises a means for copying the counting means of the plurality of blocks of code from the code cache to the storage area when the code cache is full, as shown in Figure 4, item 55 and described in the Specification beginning at paragraph [0040].

The system for analyzing a computer program that includes a plurality of blocks of code wherein the identifying means further comprises a means for determining which counting means of the plurality of blocks of code in the code cache is least recently executed and a means for copying the least recently executed block of code from the code cache to the storage area when the code cache is full, as shown in Figure 8, item 136 and described in the Specification beginning at paragraph [0051].

The system for analyzing a computer program that includes a plurality of blocks of code further comprises a means for checking the code cache to determine if the block of code is being executed for other than the first time, as shown in Figure 4, item 51 and described in the Specification beginning at paragraph [0038]. The means for loading the counting means associated with the block of code being executed for other than the first time into the counter cache, as shown in Figure 4, item 52 and described in the Specification beginning at paragraph [0038].

In accordance with Independent Claim 11, one embodiment of the present claimed invention pertains to a computer readable medium having computer-readable program code embodied therein for causing a computer system to perform a method to analyze a computer program that includes a plurality of blocks of code. As described in Figure 3 item 27 and beginning on paragraph [0036] of the Specification, a block of code is received to a code cache. A counter is used for tracking each time the block of code is executed on the code cache, as described in Figure 3, item 24 and described in the Specification beginning at paragraph [0037]. A counter cache is maintained for storing each counter of the block of code while the block of code is stored on the code cache, as shown in Figure 3, item 25 and described in the Specification beginning at paragraph [0037]. A storage area is also maintained for storing each counter of the block of code previously executed on the code cache after the block of code

is evicted from the code cache, as shown in Figure 4, item 64 and described in the Specification beginning at paragraph [0044].

The computer readable medium further comprises identifying when the code cache is full, as shown in Figure 4, item 54 and described in the Specification beginning at paragraph [0040].

The computer readable medium further comprises determining which the counter of the block of code stored on the counter cache is least recently executed and evicting the least recently executed block of code, related to the counter, from the code cache, as shown in Figure 4, item 55 and described in the Specification beginning at paragraph [0040]. In addition, the counter of the least recently executed block of code is copied from the counter cache to the storage area when the least recently executed block of code related to the counter is evicted from the code cache, as shown in Figure 8, item 136 and described in the Specification beginning at paragraph [0051].

The computer readable medium further comprises checking the storage area to determine if the block of code is being executed for other than the first time, as shown in Figure 4, item 51 and described in the Specification beginning at paragraph [0038]. The counter associated with the block of code being executed for other than the first time is then loaded from the storage area into the counter cache, as shown in Figure 4, item 52 and described in the Specification

beginning at paragraph [0038]. The counter associated with the block of code being executed for other than the first time is then updated, as shown in Figure 4, item 61 and described in the Specification beginning at paragraph [0043].

In accordance with Independent Claim 16, one embodiment of the present claimed invention pertains to a system for analyzing a computer program that includes a plurality of blocks of code. As described in Figure 3 and beginning on paragraph [0036] of the Specification, a counter that tracks each time a specific block of code is executed by a code cache, as described in Figure 3, item 24 and described in the Specification beginning at paragraph [0037]. A counter cache is provided for storing the counter of a specific block of code while the specific block of code is stored on the code cache, as shown in Figure 3, item 25 and described in the Specification beginning at paragraph [0037]. A storage area is also provided for storing the counter of a specific block of code previously executed on the code cache after the specific block of code is evicted from the code cache, as shown in Figure 4, item 64 and described in the Specification beginning at paragraph [0044].

The system for analyzing a computer program that includes a plurality of blocks of code further comprises a logic that identifies when the code cache is full, as shown in Figure 4, item 54 and described in the Specification beginning at paragraph [0040].

The system for analyzing a computer program that includes a plurality of blocks of code wherein the logic determines which counter of the specific block of code stored on the counter cache is least recently executed, evicting the least recently executed block of code related to the counter from the code cache and copies the counter of the specific block of code from the counter cache to the storage area when the least recently executed specific block of code is evicted from the code cache, as shown in Figure 8, item 136 and described in the Specification beginning at paragraph [0051].

The system for analyzing a computer program that includes a plurality of blocks of code further comprises checking the storage area to determine if the specific block of code is being executed for other than the first time and loads the counter associated with the specific block of code being executed for other than the first time, from the storage area into the counter cache and updating the counter associated with the specific block of code being executed for other than the first time, as shown in Figure 4, item 51-52 and described in the Specification beginning at paragraph [0038].

Grounds of Rejection to be Reviewed on Appeal

Claims 7-9 are rejected under 35 U.S.C. § 112, first paragraph, as failing to comply with the written description requirement.

Claims 1-2, 4-5, 11-12, 14-17 and 19-20 are rejected under 35 U.S.C. § 103(a) as unpatentable over Tremblay et al. (6,065,108, "Tremblay") in view of Burton et al. (US 6,738,865, "Burton").

Claims 6-10 are rejected under 35 U.S.C. § 103(a) as unpatentable over Holmberg et al. (2001/0021959, "Holmberg") in view of Burton.

Grouping of Claims

For each ground of rejection that applies to more than one claim, the claims do not stand or fall together. For purposes of appeal, the claims are grouped as follows:

Group 1: Claims 7-9

Group 2: Claims 1-2, 4-5, 11-12, 14-17 and 19-20

Group 3: Claims 6-10

Arguments

A. Grouping of Claims

The claims of Group 3 are considered separately patentable. The claims of Group 3 are each dependent on independent Claim 6, and each claim in Group 3 recites an additional limitation that, with the limitations of Claim 6, is patentably distinguishable over the cited art. In addition, the rejections of the dependent claims of Group 3 require additional art over the art cited against the base claim in Group 2.

B. Scope and Content of the Cited Prior Art References (Tremblay, Burton, Holmberg)

Tremblay is relied upon to disclose a method to analyze a computer program that includes a plurality of blocks of code. Including: receiving a code instruction to a code cache (cache 125, 134, 153, 155-Figure1, Figure 3-4c, Figure 6). Using a counter for tracking each time said code instruction is loaded on said code cache (cache 153-col 20, lines 14-28). Maintaining a counter cache for storing each said counter of said code instruction while said code instruction is stored on said code cache (col 20 lines 14-28).

Burton is relied upon to teach a counter for tracking block of code execution (Figure 1, counter 18).

Holmberg is relied upon to teach a method to analyze a computer program that includes a plurality of blocks of code (page 3, paragraph 0028). Executing the computer program (page 3 paragraph 0028, page 5 paragraph 0050). Using a counter for tracking each time one of said plurality of blocks is executed (access counter 25-page 3 paragraph 0034, single basic block-page 2 paragraph 0013). Maintaining a counter associated with a cache for storing said plurality of count references in a cache of said blocks of code that are most recently executed (reference 23, counter 25, Figure 1a).

C. Rejection of Claims 7-9 under 35 U.S.C. § 112, first paragraph, as failing to comply with the written description requirement. That is, the elements recited as '...said counter cache is full' (Claims 7-9, line 2, line 4, line 7, respectively) are not supported by any explicit description in the specification.

The following arguments apply to the claims of Group 1, Claims 7-9.

Claims 7-9 were amended in the response to final prior to the advisory action which was entered. The amendment to the Claims replaced the element '...said counter cache is full', 'with the element '...said code cache is full'.

For this reason, Appellant respectfully asserts that the basis for rejecting Claims 7-9 (Group 1) under 35 U.S.C. § 112, first paragraph, is overcome.

D. Rejection of Claims 1-2, 4-5, 11-12, 14-17 and 19-20 under 35 U.S.C. § 103(a) as unpatentable over Tremblay et al. (6,065,108, "Tremblay") in view of Burton et al. (US 6,738,865, "Burton").

The following arguments apply to the claims of Group 2, Claims 1-2, 4-5, 11-12, 14-17 and 19-20.

Independent Claims 1, 11 and 16 (Group 2) recite that an embodiment of the present invention includes the features of "blocks of code." In addition, embodiments include the features "using a counter for tracking each time said block of code is executed on said code cache; maintaining a counter cache for storing each said counter of said block of code while said block of code is stored on said code cache; and maintaining a storage area for storing each said counter of said block of code previously executed on said code cache after said block of code is evicted from said code cache."

With respect to Tremblay, as stated in the Final Office Action, Tremblay does not disclose the instruction cache is for storing a block of code, nor does Tremblay disclose tracking each time that said block of code is executed, nor does Tremblay disclose maintaining a storage area for storing counters associated with a block of code.

An additional statement in the Final Office Action states "if code is only a machine representation of some human-readable sequence of programming language, then programming instructions can be a block of such code, therefore, Tremblay has disclosed block of code in code cache.

However, Appellant respectfully submits that the term block of code is clearly defined in the specification including paragraph [0045] and as shown in Figure 5, a flow chart of an example of the process that translates the original source code 22 (step 56 of FIG. 4) and the placing of the translated code into the code cache 27, as shown in FIG. 4. In other words, the block of code is formed when the original source code is preprocessed to add a counter update instruction in each basic block code. In addition, the program analysis with data caching system then performs the translation of the original source and inserts this translated program code into the code cache. A new program counter (NPC), which points to the beginning of the translated code, is returned for later execution during program analysis.

Further, Tremblay actually teaches away from the utilization of blocks of code instead of the original code. Appellant understands Tremblay to teach against modifying the original instruction code to include the retrieved data and overwriting the original instruction code at the discrete location in the program memory with the modified instruction code. The modified instruction code is subsequently executed in place of the original instruction code. This is commonly

referred to as self-modifying code. One disadvantage with self-modifying code is that once the original instruction code has been modified, the original instruction code is no longer available. The unavailability of the original instruction code can hamper subsequent program debugging. Another disadvantage is that the discrete location in the memory may not have sufficient capacity to store the modified instruction code (which includes the retrieved data). In certain prior art applications, the original instruction code is stored in a read only memory (ROM). In such applications, it is not possible to overwrite the original instruction code with a modified instruction code (Tremblay column 3 beginning at line 12).

Burton is silent with respect to the features of "blocks of code." In addition, embodiments include the features "using a counter for tracking each time said block of code is executed on said code cache; maintaining a counter cache for storing each said counter of said block of code while said block of code is stored on said code cache; and maintaining a storage area for storing each said counter of said block of code previously executed on said code cache after said block of code is evicted from said code cache." Further, Burton fails to teach cache allocation based on frequency of use. Consequently, Burton does not overcome the deficiencies of Tremblay, and neither Tremblay alone or in combination with Burton et al. teach or renders obvious the features as recited in Claims 1, 11 and 16.

For this reason, Appellant respectfully asserts that the basis for rejecting Claims 1, 11 and 16 (Group 2) under 35 U.S.C. § 103(a) is overcome.

In addition, Appellant respectfully asserts that one of ordinary skill in the art combining Tremblay, teaching against the utilization of modified code and therefore no need for a counter to count the times the modified code was used, with Burton, which teaches a counter, would result in a process that utilized blocks of modified code, since Tremblay specifically teaches against modifying code for a plurality of reasons including- unavailability of the original instruction code can hamper subsequent program debugging.

For this additional reason, Appellant respectfully asserts that the basis for rejecting Claims 1, 11 and 16 (Group 2) under 35 U.S.C. § 103(a) is overcome.

To summarize, Claims 1, 11 and 16 (Group 2) recite that an embodiment of the present invention includes the features of "blocks of code." In addition, embodiments include the features "using a counter for tracking each time said block of code is executed on said code cache; maintaining a counter cache for storing each said counter of said block of code while said block of code is stored on said code cache; and maintaining a storage area for storing each said counter of said block of code previously executed on said code cache after said block of code is evicted from said code cache." Neither reference, Tremblay nor Burton, alone or in combination, teaches this element. Additionally, there is no

motivation to combine Tremblay and Burton as suggested to achieve the Claimed features. As such, Appellant respectfully submits that the basis for rejecting Claims 1, 11 and 16 (Group 2) under 35 U.S.C. § 103(a) is overcome.

Claims 2 and 4-5 (Group 2) depend from Claim 1 (Group 2). Appellant respectfully submits that the basis for rejecting Claims 2 and 4-5 under 35 U.S.C. § 103(a) is overcome as these claims depend from an allowable base claim.

Claims 12 and 14-15 (Group 2) depend from Claim 11 (Group 2). Appellant respectfully submits that the basis for rejecting Claims 12 and 14-15 under 35 U.S.C. § 103(a) is overcome as these claims depend from an allowable base claim.

Claims 17 and 19-20 (Group 2) depend from Claim 16 (Group 2). Appellant respectfully submits that the basis for rejecting Claims 17 and 19-20 under 35 U.S.C. § 103(a) is overcome as these claims depend from an allowable base claim.

E. Rejection of Claims 6-10 under 35 U.S.C. § 103(a) as unpatentable over Holmberg et al. (2001/0021959, "Holmberg") in view of Burton.

The following arguments apply to the claims of Group 3, Claims 16-10.

Independent Claims 6 (group 3) recite that an embodiment of the present invention includes the features of " a means for counting each time one of said plurality of blocks of code is executed; means for maintaining a counter cache for storing said counting means of said plurality of blocks of code that are most recently executed; and means for maintaining a storage area for storing said counting means of said plurality of blocks of code that are most recently executed."

With respect to Holmberg, as stated in the Office Action, Holmberg does not disclose maintaining a counter cache for storing a counter associated with a block of code, nor does Holmberg disclose maintaining a storage area for storing counters associated with a block of code.

Further, Appellant understands Holmberg to teach against cache allocation based only on frequency of use. That is, Appellant understands Holmberg to teach the utilization of additional features for cache allocation such as importance of the instruction. For example, Applicant understands Holmberg to teach that the processor can do measurements

that do not count accesses from these program blocks or discard accesses made from programs running on lower priority levels. Thus, Holmberg actually teaches away from the utilization of frequency for programs running on lower priority levels (paragraph [0054]. Therefore, Holmberg neither teaches nor render obvious the features of the present invention.

Burton is silent with respect to the features of "a means for counting each time one of said plurality of blocks of code is executed; means for maintaining a counter cache for storing said counting means of said plurality of blocks of code that are most recently executed; and means for maintaining a storage area for storing said counting means of said plurality of blocks of code that are most recently executed." Further, Burton fails to teach cache allocation based on frequency of use. Consequently, Burton does not overcome the deficiencies of Holmberg, and neither Holmberg alone or in combination with Burton et al. teach or renders obvious the features as recited in Claims 1, 11 and 16.

For this reason, Appellant respectfully asserts that the basis for rejecting Claim 6 (Group 2) under 35 U.S.C. § 103(a) is overcome.

To summarize, Claim 6 (Group 3) recites that an embodiment of the present invention includes the feature of "means for maintaining a counter cache for storing said counting means of said plurality of blocks of code that are most

recently executed." Neither reference, Holmberg nor Burton, alone or in combination, teaches this element. As such, Appellant respectfully submits that the basis for rejecting Claim 6 (Group 3) under 35 U.S.C. § 103(a) is overcome.

Claim 7-10 (Group 3) depend from Claim 6 (Group 3). Appellant respectfully submits that the basis for rejecting Claims 7-10 under 35 U.S.C. § 103(a) is overcome as this claim depends from an allowable base claim.

Conclusion

Appellant believes that pending Claims 1, 2, 4-12, 14-17, 19 and 20 are patentable over the cited art. Appellant respectfully requests that the rejection of these claims be reversed.

Respectfully submitted,

WAGNER, MURABITO & HAO LLP

Date: 6/15/05

John P. Wagner
Registration Number: 35,398

WAGNER, MURABITO & HAO LLP
Two North Market Street
Third Floor
San Jose, CA 95113
408-938-9060

Appendix - Clean Copy of Claims

1. (previously presented) A method to analyze a computer program that includes a plurality of blocks of code, the method comprising:

receiving a block of code to a code cache;

using a counter for tracking each time said block of code is executed on said code cache;

maintaining a counter cache for storing each said counter of said block of code while said block of code is stored on said code cache; and

maintaining a storage area for storing each said counter of said block of code previously executed on said code cache after said block of code is evicted from said code cache.

2. (previously presented) The method of Claim 1, further comprising the step of:

identifying when said code cache is full.

3. (canceled)

4. (previously presented) The method of Claim 2, further comprising:

determining which said counter of said block of code stored on said counter cache is least recently executed; and

evicting said least recently executed block of code, related to said counter, from said code cache; and

copying said counter of said least recently executed block of code from said counter cache to said storage area when said least recently executed block of code related to said counter is evicted from said code cache.

5. (previously presented) The method of Claim 1, wherein said receiving a block of code to a code cache further comprises:

checking said storage area to determine if said block of code is being executed for other than the first time;

loading said counter associated with said block of code being executed for other than the first time, from said storage area into said counter cache; and

updating said counter associated with said block of code being executed for other than the first time.

6. (original) A system for analyzing a computer program that includes a plurality of blocks of code, comprising:

means for executing said computer program;

means for counting each time one of said plurality of blocks of code is executed;

means for maintaining a counter cache for storing said counting means of said plurality of blocks of code that are most recently executed; and

means for maintaining a storage area for storing said counting means of said plurality of blocks of code that are most recently executed.

7. (previously presented) The system of Claim 6, further comprising:

means for identifying when said code cache is full.

8. (previously presented) The system of Claim 7, further comprising:

means for copying said counting means of said plurality of blocks of code from said code cache to said storage area when said code cache is full.

9. (previously presented) The system of Claim 8, wherein said identifying means further comprises:

means for determining which said counting means of said plurality of blocks of code in said code cache is least recently executed; and

means for copying said least recently executed block of code from said code cache to said storage area when said code cache is full.

10. (previously presented) The system of Claim 8, further comprising:

means for checking a code cache to determine if a block of code is being executed for other than the first time; and

means for loading said counting means associated with said block of code being executed for other than the first time, into said counter cache.

11. (previously presented) A computer readable medium having computer-readable program code embodied therein for causing a computer system to

perform a method for analyzing a computer program that includes a plurality of blocks of code comprising:

receiving a block of code to a code cache;

utilizing a counter for tracking each time said block of code is executed on said code cache;

maintaining a counter cache for storing each said counter of said block of code while said block of code is stored on said code cache; and

maintaining a storage area for storing each said counter of said block of code previously executed on said code cache after said block of code is evicted from said code cache.

12. (previously presented) The computer readable medium of Claim 11, further comprising:

identifying when said code cache is full.

13. (canceled)

14. (previously presented) The computer readable medium of Claim 12, further comprises:

determining which said counter of said block of code in said counter cache is least recently executed;

evicting said least recently executed block of code, related to said counter, from said code cache; and

copying said counter of said least recently executed block of code from said counter cache to said storage area when said least recently executed block of code related to said counter is evicted from said code cache.

15. (previously presented) The computer readable medium of Claim 13, wherein said receiving a block of code to a code cache further comprises:

checking said storage area to determine if said block of code is being executed for other than the first time;

loading said counter associated with said block of code being executed for other than the first time, from said storage area into said counter cache; and

updating said counter associated with said block of code being executed for other than the first time.

16. (previously presented) A system for analyzing a computer program that includes a plurality of blocks of code, the system comprising:

a counter that tracks each time a specific block of code is executed by a code cache;

a counter cache for storing said counter of a specific block of code while said specific block of code is stored on said code cache; and

a storage area for storing said counter of a specific block of code previously executed on said code cache after said specific block of code is evicted from said code cache.

17. (previously presented) The system of Claim 16, further comprising:

logic that identifies when said code cache is full.

18. (canceled)

19. (previously presented) The system of Claim 17, wherein said logic determines which said counter of said specific block of code stored on said counter cache is least recently executed, evicting said least recently executed block of code related to said counter from said code cache, and copies said counter of said specific block of code from said counter cache to said storage area when said least recently executed specific block of code is evicted from said code cache.

20. (previously presented) The system of Claim 17, wherein said logic checks said storage area to determine if said specific block of code is being executed for other than the first time, and loads said counter associated with said specific block of code being executed for other than the first time, from said storage area into said counter cache, and updating said counter associated with said specific block of code being executed for other than the first time.